

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期： 2019.11.3**

**网络空间安全学院**

**目 录**

**[3 函数与程序结构实验 1](#_Toc404837920)**

[3.1 实验目的 1](#_Toc404837921)

[3.2 实验内容 1](#_Toc404837922)

[3.3 实验小结 1](#_Toc404837923)

# 3 函数与程序结构实验

## 3.1 实验目的

（1）熟悉和掌握函数的定义、声明；函数调用与参数传递方法；以及函数返回值类型的定义和返回值使用。

（2）熟悉和掌握不同存储类型变量的使用。

（3）熟悉多文件编译技术。

## 3.2 实验内容

**3.2.1 源程序改错**

下面是计算s=1!+2!+3!+…+n!的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

#include "stdio.h"

void main(void)

{

int k;

for(k=1;k<6;k++)

printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));

}

long sum\_fac(int n)

{

long s=0;

int i;

long fac;

for(i=1;i<=n;i++)

fac\*=i;

s+=fac;

return s;

}

**解答：** （1）错误修改：

1) 第1行的头文件定义错误，正确形式为：

#include<stdio.h>

2) 第3行的voidmain( void )，正确形式为：

main( void )

3）第11行未定义i的初始值，正确形式为：

i=0;

4）第13行的for循环后应有花括号，正确形式为：

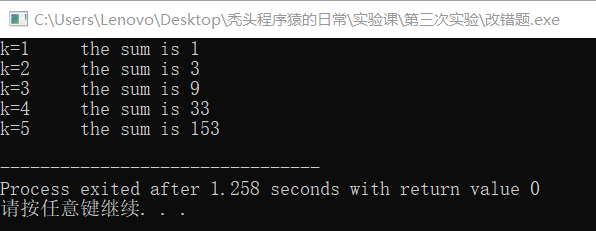
for(i=1;i<=n;i++)

{fac\*=i;

s+=fac;}

5）第16行 return 和花括号的位置放反了

（2）错误修改后运行结果：



**图3-1 改错题的运行结果**

**3.2.2 源程序修改替换**

（1）修改第1题中sum\_fac函数，使其计算量最小。

（2）修改第1题中sum\_fac函数，计算。

**解答：**

（1）如果每次计算 n!都从头计算，计算量显然很大，可以利用上一次计算的结果，

使用 static 变量，另外计算 s 的值的时候也可以利用上一次计算的结果。 #include<stdio.h>

long sum\_fac(int);

long fac(int);

static long m=1;

static long s =0;

int main(void) {

int k;

for (k = 1; k <= 6; k++)

printf("k=%d\tthe sum is %ld\n", k, sum\_fac(k));

return 0; }

long sum\_fac(int n)

{

s += fac(n);

return s; }

29

long fac(int k) { m \*= k; return m; }

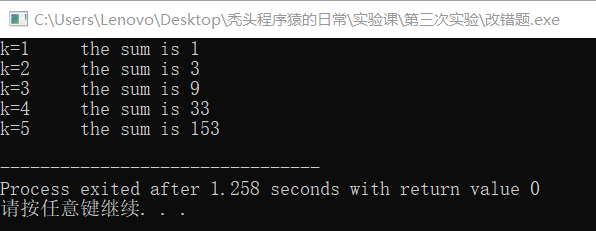


图3-2 程序修改替换题运行结果

（2）本程序不只需要加语句 fac=1/fac，还需要将所有 long 改成 double

#include<stdio.h>

double sum\_fac(double n)

{

double s=0.0;

int i;

double fac=1.0;

for(i=1;i<=n;i++)

{fac\*=(1/((double)i));

s+=fac;}

return s;

}

int main(void)

{

int k;

for(k=1;k<6;k++)

printf("k=%d\tthe sum is %lf\n",k,sum\_fac(k));

}

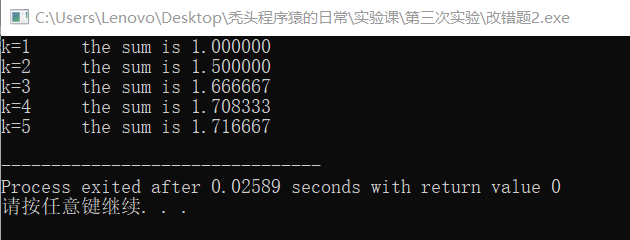


图3-3 程序修改替换题运行结果

**3.3.3跟踪调试题**

下面是计算fabonacci数列前n项和的源程序，现要求单步执行该程序，观察p,i,sum,n值，即：

（1）刚执行完scanf("%d",&k);语句，p,i值是多少？

（2）从fabonacci函数返回后光条停留在哪个语句上？

（3）进入fabonacci函数时，watch窗口显示的是什么？

（4）当i=3时，从调用fabonacci函数到返回，n值如何变化？

**源程序**

void main(void)

{

int i,k;

long sum=0,\*p=&sum;

scanf("%d",&k);

for(i=1;i<=k;i++){

sum+=fabonacci(i);

printf("i=%d\tthe sum is %ld\n",i,\*p);

}

}

long fabonacci(int n)

{

if(n==1 || n==2)

return 1;

else

return fabonacci(n-1)+fabonacci(n-2);

}

其中，long sum=0,\*p=&sum;声明p为长整型指针并用&sum取出sum的地址对p初始化。\*p表示引用p所指的变量（\*p即sum）。

**解答：**

1. p,i的值如图所示

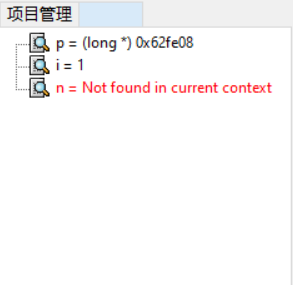


图3-4 跟踪调试题图1

（2）光条停留在如图所示的语句上：

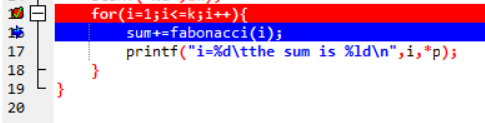


图3-5跟踪调试题图2

（3）watch窗口显示如图所示：

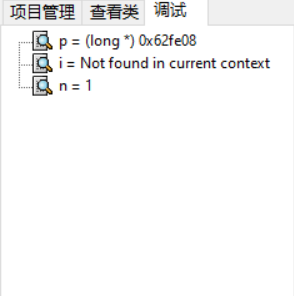


图3-6跟踪调试题图3

（4）n的值由 3 变为 2，又变为 3，再变为 1，再变为 3。

**3.3.4．编程设计题**

（1）编程让用户输入两个整数，计算两个数的最大公约数并且输出之（要求用递归函数实现求最大公约数）。同时以单步方式执行该程序，观察递归过程。

1. 算法流程如图所示：

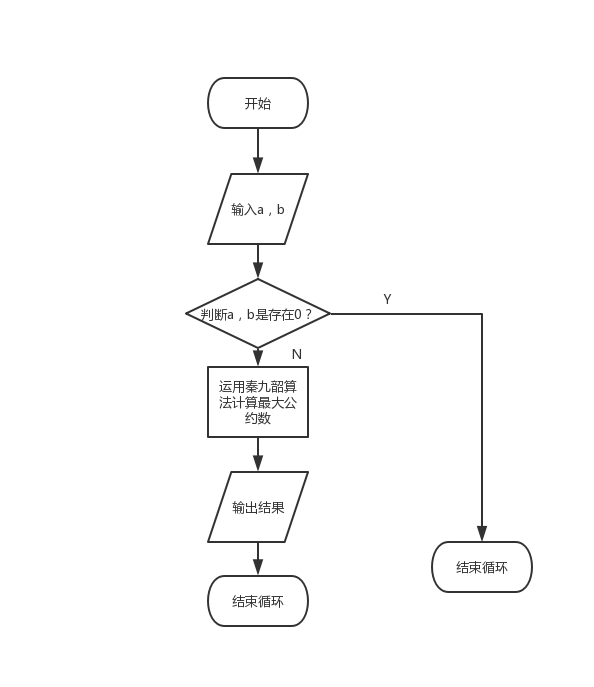


图3-7 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

int main()

{

int m,n,q,a,b,c;

printf("请输入两个整数：");

scanf("%d%d",&a,&b);

if ((a==0)&&(b==0))

return 0;

else

{ m = a;

n = b;

if (n > m)

{

int z;

z = m;

m = n;

n = z;

}

do

{

q = m%n;

m = n;

n = q;

}

while (q != 0);

printf("%d ",m );

}

} 3）测试

（a） 测试数据：

9 27

81 27

（b） 对应测试数据的运行结果截图

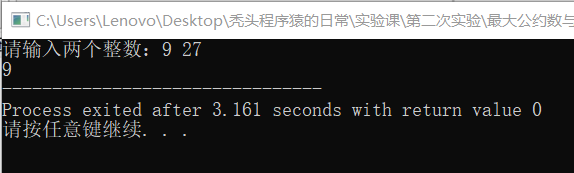


图3-8 编程题1的运行结果实例1

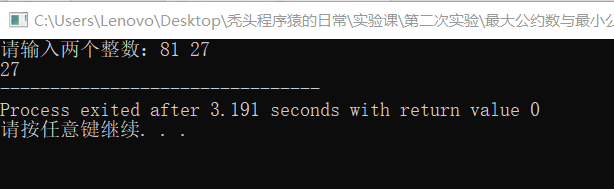


图3-9编程题1的运行结果实例2

（2）编程验证歌德巴赫猜想：一个大于等于4的偶数都是两个素数之和。

**解答：**

1. 解题思路：

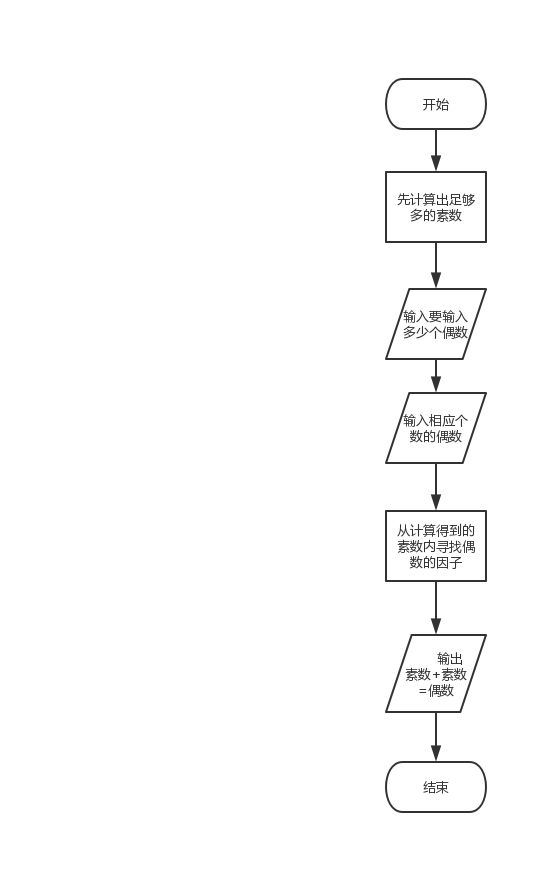


图3-10 编程题2的程序流程图

2）程序清单：

#include<stdio.h>

int main(void)

{

int i,j,a[100],b[100];

for (a[0]=a[1]=0,i=2;i<100;i++)

a[i]=1;

for (i=2;i<50;i++)

if(a[i])

{for(j=i\*i;j<100;j+=i)

a[j]=0;

}

for(i=2,j=0;i<100;i++)

if(a[i]){

b[j]=i;

j++;

}

int p,w[100],x,y,z;

scanf("%d",&x);

for(y=0;y<100;y++)

w[y]=0;

for(y=0;y<x;y++)

scanf("%d",&w[y]);

i=0;

inx:

if (w[i]!=0)

{

for(y=0;y<25;y++)

{z=w[i]-b[y];

for(p=0;p<25;p++)

{

if (z==b[p])

{printf("%d=%d+%d\n",w[i],b[y],z);

i++;

goto inx;

}

}

}

}

} 3）测试

（a） 测试数据：

测试数据的方法如表1-1所示

表3-1 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 20 | 20=3+17 | 20=3+17 |
| 用例2 | 24 | 24=5+19 | 24=5+19 |
| 用例3 | 10 | 10=3+7 | 10=3+7 |

（b）运行结果：

对应测试运行结果如图所示。

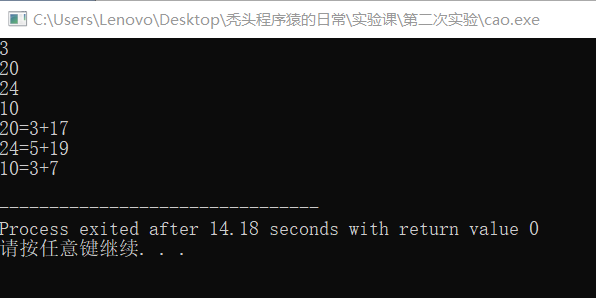


图3-11编程题2的运行结果实例1

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**（3）**编写一个程序，证明对于在符号常量BEGIN和END之间的偶数这一猜测成立。例如，如果BEGIN为10，END为20，程序的输出应为：

GOLDBACH'S CONJECTURE:

Every even number n>=4 is the sum of two primes.

10=3+7

12=5+7

……

20=3+17

**解答：**

1. 算法流程如上图所示
2. 源程序清单

#include<stdio.h>

#define BEGIN 10

#define END 20

int main(void)

{

int i,j,a[100],b[100];

for (a[0]=a[1]=0,i=2;i<100;i++)

a[i]=1;

for (i=2;i<50;i++)

if(a[i])

{for(j=i\*i;j<100;j+=i)

a[j]=0;

}

for(i=2,j=0;i<100;i++)

if(a[i]){

b[j]=i;

j++;

}

int p,w[100],x,y,z;

x=END-BEGIN+1;

for(y=0;y<100;y++)

w[y]=0;

for(y=10;(y>=BEGIN)&&(y<=END);y++)

w[y]=y;

i=10;

printf("GOLDBACH'S CONJECTURE:\nEvery even number n>=4 is the sum of two primes.\n");

inx:

if ((w[i]!=0)&&((w[i])%2==0)&&(w[i]<=20))

{

for(y=0;y<25;y++)

{z=w[i]-b[y];

for(p=0;p<25;p++)

{

if ((z==b[p])&&(z!=2))

{printf("%d=%d+%d\n",w[i],b[y],z);

i++;

goto inx;

}

}

}

}

else

{

i++;

goto inx;

}

}

1. 测试

测试结果如图所示：

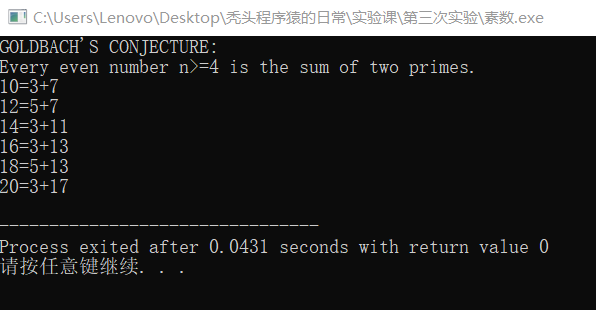


图3-12 编程题3的运行结果实例

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**5．选做题**

1）程序代码：

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);

func1();

return 0;

}

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}

#include <stdio.h>

extern int x,y; /\* 外部变量的引用性说明 \*/

extern char ch; /\* 外部变量的引用性说明 \*/

void func1(void); /\* func1函数原型 \*/

1. 运行结果如图：

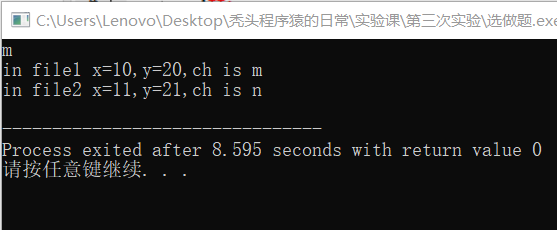


图3-13 选作题运行结果

## 1.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。  
 在这次实验的过程中，我在进行选做题实验时曾经遇到过不会正确链接多个程序的问题。最终我通过咨询同学解决了这一问题。

通过本次实验课，我深刻地体会到了“纸上得来终觉浅，绝知此事要躬行”这句话所蕴含的深刻道理。也许课上听完讲后我觉得自己已经掌握了相关的知识，但直到自己亲自动手编程是才发现自己的能力并没有想象中那么出色。希望我以后可以在这种活动中收获更多！

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计,北京： 科学出版社,2013

[2] 李开,卢萍,曹计昌. C语言实验与课程设计, 北京：科学出版社,2011